

# Butterfly Effect: Peeling Bipartite Networks

Ahmet Erdem Sarıyüce  
Sandia National Laboratories  
Livermore, CA, USA  
asariyu@sandia.gov

Ali Pinar  
Sandia National Laboratories  
Livermore, CA, USA  
apinar@sandia.gov

## ABSTRACT

Affiliation, or two-mode, networks, such as actor-movie, document-keyword, or user-product are prevalent in a lot of applications. The networks can be most naturally modeled as bipartite graphs, but most graph mining algorithms and implementations are designed to work on the classic, unipartite graphs. Subsequently, studies on affiliation networks are conducted on the co-occurrence graphs (e.g., co-authors and co-purchase networks), which projects the bipartite structure to a unipartite structure by connecting two entities if they share an affiliation. Despite their convenience, co-occurrence networks come at a cost of loss of information and an explosion in graph sizes. In this paper, we study the dense subgraph discovery problem on bipartite graphs. We propose peeling algorithms to find many dense substructures and a hierarchy among them. Our peeling algorithms are based on the butterfly subgraphs (2,2-bicliques). Experiments show that we can identify much denser structures compared to the state-of-the-art techniques on co-occurrence graphs. Our algorithms are also memory efficient, since they do not suffer from the explosion in the number of edges of the co-occurrence graph. An in-depth analysis on the author-paper network of the top database conferences highlights the richer structure that can be identified by working on bipartite graphs, which is otherwise lost in a co-occurrence network.

## 1. INTRODUCTION

Many real-world systems are naturally modeled as affiliation, two-mode or bipartite networks [12, 31]. In a bipartite network, nodes can be divided into two partitions, and the edges connect only pairs from different partitions. For example authors and papers can be two vertex partitions and an edge representing authorship. Or actors and movies for two partitions where an edge means the actor played in the corresponding movie. The vertex partition that drives the network formation is the *primary* set, and the other partition is the *secondary* set. For instance the authors set

is primary, and the papers set is secondary in the authorship networks. Despite their representation power, bipartite graphs are underutilized in many applications of network analysis, since most network problems are studied on traditional unipartite co-occurrence graphs. Instead of exploiting the accurate bipartite representation, co-occurrence graphs are constructed, such that two nodes are connected by an edge if they share an affiliation. For instance, an author-paper network can be transformed into a co-authorship network, where two authors are connected if they co-authored paper. However, this transformation comes at a cost of information loss and increased graph sizes, as we will discuss in more detail later. Therefore, designing algorithms that can work directly on the bipartite data, which provides an accurate representation of underlying system, is necessary.

In this paper, we study the problem of finding dense structures in a *bipartite* graph. Finding dense subgraphs in real-world networks, and relating them to each other has been shown to be useful across different domains. Literature is rich with the benefits: spam group detection in web [22], migration pattern discovery in stock market [19], gene co-expression network analysis [50], real-time story identification in microblogging websites [6], and throughput improvement of social-networking sites [23] are just a few examples.  $k$ -core [47, 35] and  $k$ -truss [44, 17] decompositions are peeling algorithms and have been shown to be a simple way to extract the structure of unipartite graphs in a regularized way. When applied on higher-order structures (triangles or small size cliques), many high-quality dense subgraphs can be obtained with detailed hierarchical relations [46, 28].

### 1.1 Problem and Challenges

We focus on simple undirected bipartite graphs. Our aim is to find many dense regions and determine the relations among them by using a higher-order peeling algorithm.

**Why bipartite?:** In order to work on bipartite networks, a common practice in the literature has been creating the co-occurrence graph, which is also known as the projection. There also exists weighted projections that assigns weights on the edges to better reflect the bipartite nature of the data [38, 39]. Although the projection enables the use of well-studied unipartite graph mining algorithms [12], it has some drawbacks:

- **Information loss and distortion:** Bipartite network has the information on one-to-many relationships, but it is reduced to pairwise ties when projected to a weighted or unweighted unipartite form. Furthermore,

those pairwise ties are independently represented and processed which distorts the original information.

- **Ambiguity:** Projections are not bijective irrespective of the conversion technique being used. For instance, any two bicliques with equal number of primary vertices result in the same unweighted projection.
- **Edge inflation:** Each secondary vertex in the bipartite network with degree  $d_i$  results in a  $d_i$ -clique in the projected graph. Thus, the number of edges in the projected graph can be as many as  $\sum_{v \in V} \binom{d_v}{2}$ , whereas it is only  $\sum_{v \in V} d_v$  in the bipartite network, where  $V$  is the set of secondary vertices. Increasing size reduces the performance and also artificially boosts the clustering coefficients and local density measures in the projected graph.

Motivated by the incapability and inefficiency of projection approaches, we work directly on the bipartite graph to discover the dense structures in a regularized way.

**Why peeling?:**  $k$ -core and  $k$ -truss decomposition algorithms have been shown to be effective to find the dense regions and determining relations among them [28, 51, 24]. Furthermore, they are used as a building block to efficiently solve NP-Hard problems like maximal clique finding [7, 26]. However, they are not directly applicable for bipartite networks.  $k$ -core decomposition assumes that all the vertices in the graph represent the same kind of entity, but it is not the case in the bipartite graphs. Main focus is always on one of the vertex sets (mostly primary) and there are no edges among them. Regarding the  $k$ -truss, the problem is that there is no triangle. It has been shown that the higher-order structures offer deeper insights for analyzing real-world networks and detecting the dense regions in a better way [46, 10, 28]. Thus, it is essential to define new higher-order structures that capture the triangle-like semantic in bipartite graphs, which would enable the peeling adaptations.

## 1.2 Contributions

We introduce higher-order peeling algorithms for bipartite graphs. Our contributions can be summarized as follows:

- **Higher-order bipartite structures:** We survey the existing attempts to define higher-order structures in bipartite graphs, and select the **butterfly** structure (2,2-biclique) as the simplest super-edge motif. Building on that, we define the  **$k$ -tip** and  **$k$ -wing** subgraphs based on the involvements of vertices and edges in butterflies, respectively.
- **Butterfly peeling algorithms:** In order to efficiently find all the  $k$ -tips and  $k$ -wings, we introduce butterfly peeling algorithms. Our algorithms are iterative in nature, and inspired by the degeneracy based decompositions for unipartite graphs. We also analyze their complexity, and compare with the alternatives.
- **Experimental evaluation:** We evaluate our new definitions and algorithms on real-world networks. We first find the dense subgraph profiles for our contributions as well as the alternatives in the literature. Figure 1 gives a glance of results on the movie-actor network from IMDb. Our algorithms are able to extract denser subgraphs of different sizes. We also apply our algorithms on the author-paper network of

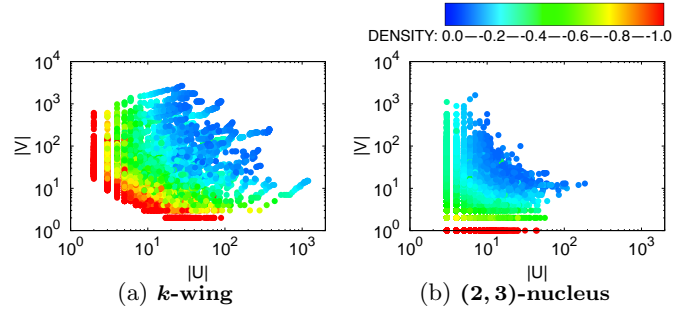


Figure 1: Dense subgraph profile of IMDb network. Density is color coded and size of each vertex partition is given by the x- and y-axis. Our wing decomposition algorithm results in 36 bipartite subgraphs that have at least 10 vertices in each side with  $\geq 0.9$  edge density. Competitor algorithms that work on projections are not able to give such dense bipartite structures.

top database conferences, and highlight the interesting subgraphs and hierarchies we detect. Lastly, we present the runtime performance of our peeling algorithms and compare with the existing alternatives.

## 2. BACKGROUND

This section reminds the existing definitions about bipartite networks and the peeling algorithms, and presents our notations.

Let  $G = (U, V, E)$  be an undirected unweighted simple (no loop, no multi-edge) bipartite graph.  $U$  is the set of primary vertices,  $V$  is the set of secondary vertices, and  $E$  is the set of edges s.t.  $\forall (u, v) \in E, u \in U \wedge v \in V$ .  $N(u)$  denotes the neighborhood of a vertex s.t. if  $u \in U$ ,  $N(u) = \{v \mid (u, v) \in E\}$  (similarly defined for  $v \in V$ ).  $G = (U, V, E)$  is an **(a,b)-biclique** if  $\forall u \in U$  and  $\forall v \in V$ ,  $\exists (u, v) \in E$  and  $|U| = a, |V| = b$ . Here, we give two ways to convert the bipartite graph to the unipartite graph, both of which are also illustrated in Figure 2b:

**DEFINITION 1. Unweighted projection** of  $G = (U, V, E)$  is the unipartite graph  $G_p = (V_p, E_p)$  s.t.  $V_p = U, E_p = \{(u_1, u_2) \mid N(u_1) \cap N(u_2) \neq \emptyset\}$ .

**DEFINITION 2. Weighted projection** of  $G = (U, V, E)$  is the unipartite graph  $G_{wp} = (V_{wp}, E_{wp})$  s.t.  $V_{wp} = U$ ,  $E_{wp} = \{(u_1, u_2, w) \mid N(u_1) \cap N(u_2) \neq \emptyset \wedge w = \sum_{v \in N(u_1) \cap N(u_2)} \frac{1}{|N(v)|}\}$  [38, 39].

$k$ -core [47, 35] and (2,3)-nucleus [46, 28] definitions are as follows:

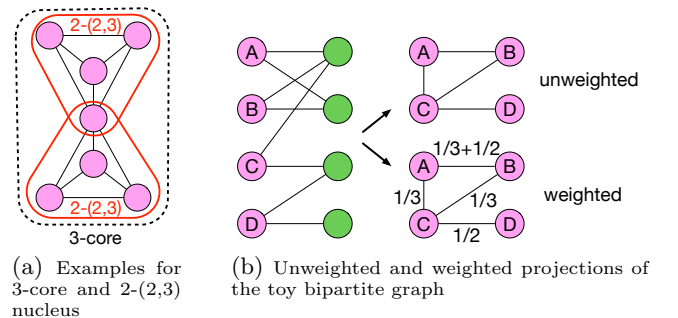


Figure 2: Illustrative examples for Definitions 1, 2, 3, and 4

DEFINITION 3. A connected and maximal subgraph  $H$  is  **$k$ -core** if every vertex in  $H$  has at least degree  $k$ .

DEFINITION 4. Subgraph  $H = (V, E)$  is a  **$k$ -(2,3)-nucleus**, if (1) for every edge  $(u, v) \in E$ ,  $|N(u) \cap N(v)|$  is at least  $k$ , (2) any pair of edges in  $E$  is connected by series of triangles, (3) it is maximal (no other edge can be added).

Figure 2a gives an example. Entire graph is a 3-core since the minimum degree is 3. There is no other  $k$ -core. However, there exists two 2-(2,3) nuclei, which are also 4-cliques. Note that each edge in the 2-(2,3) nucleus is contained in two triangles. We do not combine two 2-(2,3) nuclei, because there is no triangle that binds those two structures, violating the connectivity condition (2) in Definition 4. More details about nucleus decompositions can be found in [46].

### 3. HIGHER-ORDER BIPARTITE STRUCTURES

Capturing the smallest unit of cohesion provides a structural way to find the dense regions. In the literature, different triangle-like structures [12, 43, 42] and density measures [31] have been proposed for bipartite graphs, which are also illustrated in Figure 3. Borgatti and Everett considered that **(3,3)-biclique** is the best smallest structure to represent cohesiveness [12]. Their reasoning is that it is the smallest subgraph that results in a triangle when projected to the unipartite graph. Opsahl had the same motivation [42], but he proposed to use the **closed 4-path** in a bipartite graph to create a triangle in its projection. He defines the 4-path as a four connected edges between three primary and two secondary vertices, and it is **closed** if there is a third secondary vertex that is connected to the two primary vertices which had only one neighbor in the 4-path. Density is regarded as the ratio of the closed 4-paths to all 4-paths. This is more relaxed than the (3,3)-biclique of [12], and is able to create a triangle in the projection with less number of edges. Robins and Alexander considered the problem without any projections [43]. They suggested to use the (2,2)-biclique to model the cohesion, since it is the simplest and most local form of cycle. Similar to the Opsahl’s work, they looked for a 3-path, which consists of three edges with two primary and two secondary vertices, and defines the density as the ratio of closed 3-paths ((2,2)-bicliques) to the all 3-paths. This approach is also embraced in a recent work by Aksoy *et al.* [3] to generate bipartite graphs with community structure. They named the 3-path as a **caterpillar**, and the (2,2)-biclique as a **butterfly**. Latapy *et al.* [31] approached the problem from a different angle and proposed to look at the Jaccard similarities of the neighbor lists of a primary vertex pair to measure the density. Alternatively, they also suggested to use the Overlap similarity to capture the vertices that resides in the overlap of multiple dense regions, where the denominator is the smaller set size.

We use the butterfly as the higher-order structure since it is the smallest unit of cohesion in bipartite graphs. It is the smallest structure with multiple vertices at each side, and also cheaper to enumerate than the larger bicliques.

#### 3.1 $k$ -tip

Inspired by the  $k$ -core and nucleus decomposition concepts, we focus on the involvements of the vertices and edges in the higher-order structures of bipartite graphs. For this purpose, we first define the  **$k$ -tip** bipartite subgraph:

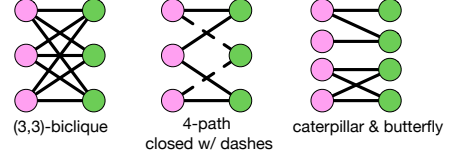


Figure 3: Higher-order bipartite structures in the literature

DEFINITION 5. Bipartite subgraph  $H = (U, V, E)$  is an **induced subgraph** on set  $U$ , s.t.

$$V = \bigcup_{u \in U} N(u), \quad E = \bigcup_{u \in U} \bigcup_{v \in N(u)} (u, v).$$

$H$  is a  **$k$ -tip** if

- every vertex  $u \in U$  takes part in at least  $k$  butterflies,
- each vertex pair  $u_1, u_2 \in U$  are connected by a set of butterflies,
- it is maximal, i.e., there is no  $H' \supset H$  where  $H'$  is a  $k$ -tip.

Involvement of vertices in butterflies is achieved by the number of incidences (1), and the connectivity condition (2), where two vertices are connected if they share a butterfly directly, or transitively by some other intermediate vertices. In addition, we define the **tip number** of a vertex as follows:

DEFINITION 6. Tip number of  $u \in U$ , denoted as  $\theta(u)$ , is  $t$  if there is a  $t$ -tip that includes  $u$ , and there is no  $t'$ -tip that contains  $u$  s.t.  $t' > t$ .  $H = (U, V, E)$  is a  **$k$ -tip** when  $\forall u \in U, \theta(u) \geq k$ .

Figure 4a illustrates some  $k$ -tips on a toy graph. Vertices on the left are primary ( $U$ ) (1-6), and the others are secondary ( $V$ ) (a-f). In total there are 9 butterflies: 1a2b, 2a3b, 2a3c, 2b3c, 3c4d, 4d5e, 4d5f, 4e5f, and 5e6f. Vertices 1 and 6 take part in only one butterfly. All the others in  $U$  are involved in more than one butterfly. Thus, the entire graph is a 1-tip, shown with blue. If we only take the subgraph induced by vertices 2, 3, 4 and 5, we see a different picture. Butterfly count of vertices 2 and 5 are three and 3 and 4 are four. So, this subgraph is a 3-tip and shown in red region. Note that all the vertices in the 3-tip are connected to each other by butterflies. We say  $\theta(1)=\theta(6)=1$  (blue nodes) since they are only contained in the 1-tip, and other vertices have  $\theta=3$  (red nodes). Algorithm to find the tip number of vertices, and all the  $k$ -tips is given in Section 4.

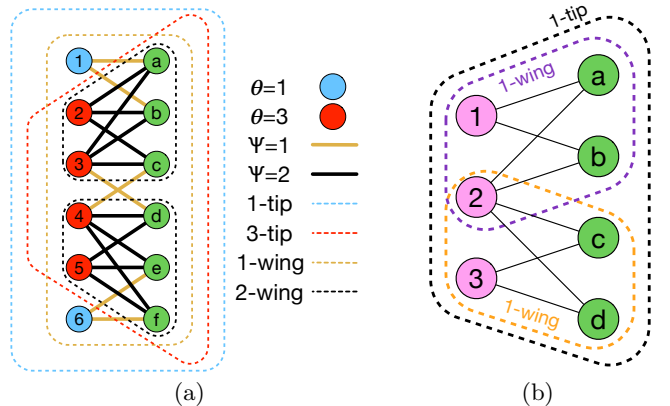


Figure 4: Illustration of  $k$ -tip and  $k$ -wing subgraphs

### 3.2 $k$ -wing

When finding the dense regions, or detecting the communities, high degree vertices cannot be in a single subgraph. Multiple dense regions overlap on those vertices. Given the importance of overlaps, we focus on the relations between edge and butterfly structures. We define  $k$ -wing and **wing number** as follows:

DEFINITION 7. *Bipartite subgraph  $H = (U, V, E)$  is a  $k$ -wing if*

- every edge  $(u, v) \in E$  takes part in at least  $k$  butterflies,
- each edge pair  $(u_1, v_1), (u_2, v_2) \in E$  is connected by series of butterflies,
- it is maximal (no other edge can be added).

DEFINITION 8. *Wing number of  $(u, v) \in E$ , denoted as  $\psi(u, v)$ , is  $w$  if there is a  $w$ -wing that includes  $(u, v)$ , and there is no  $w'$ -wing that contains  $(u, v)$  s.t.  $w' > w$ . If  $H = (U, V, E)$  is a  $k$ -wing, then  $\forall (u, v) \in E$ ,  $\psi(u, v) \geq k$ .*

A simple example is given in Figure 4b. Each edge has only one butterfly. However, there is not one, but two separate 1-wings, because there is no butterfly that contains both  $(2, b)$  and  $(2, c)$ .  $k$ -wing is able to distinguish the  $(2, 2)$ -biclique  $(1, 2, a, b)$  from the other  $(2, 2)$ -biclique  $(2, 3, c, d)$ . Also entire graph is a 1-tip, and there is no denser  $k$ -tip.

We also find the  $k$ -wings in the toy graph given in Figure 4a. Edges with the minimum number of butterflies are  $(1, a), (1, b), (3, d), (4, c), (6, e)$  and  $(6, f)$ , shown in gold. Each has only one butterfly, so the entire graph is a 1-wing. There are also two  $(2, 3)$ -bicliques;  $(2, 3, a, b, c)$  and  $(4, 5, d, e, f)$ , denoted by black. Each of them is a 2-wing because all their edges have two butterflies, for example  $(2, a)$  has  $2a3b$  and  $2a3c$ . Thus the wing numbers of the edges in the 2-wings are 2, and others are 1.

Note that,  $k$ -wing definition is able to find the subgraphs with high pairwise overlap similarities, as suggested by Latapy *et al.* [31]. If we measure the density of a  $k$ -wing which contains a high degree vertex, it is likely to have low pairwise Jaccard similarities, but the overlap similarities will be high.

## 4. PEELING BUTTERFLIES

In this section, we give algorithms to find all the  $k$ -tips and  $k$ -wings in a bipartite graph. Our algorithms find the  $\theta(\cdot)$  and  $\psi(\cdot)$  numbers of vertices and edges, respectively, and also builds the  $k$ -tip and  $k$ -wing subgraphs on-the-fly with the hierarchical relations among them.

### 4.1 Tip Decomposition

We leverage the degeneracy-like pattern to find the tip numbers of vertices. We aim to detect how much each vertex is involved in butterflies. Our algorithm is similar to the  $k$ -core decomposition. We start with counting the number

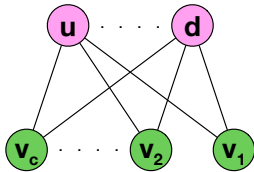


Figure 5: We first combine the distance-2 neighbors  $u$ , then find the duplicate counts,  $c$ , for all vertices. Summation of  $\binom{c}{2}$ s gives the butterfly count of  $u$

---

#### Algorithm 1: TIPDECOMPOSITION( $G = (U, V, E)$ )

---

```

// find the number of  $\mathbf{X}$  for each vertex  $u \in U$ 
1  $D \leftarrow \text{list}, L \leftarrow \text{list}, \beta(\cdot) \leftarrow \text{butterfly counts } \forall u \in U$ 
2 for each  $u \in U$  do
3    $D \leftarrow \text{combine } N(v), \forall v \in N(u)$ 
4    $C \leftarrow \text{duplicate counts of all } d \in D \text{ s.t. } d \neq u$ 
5    $\beta(u) \leftarrow \sum_{c \in C} \binom{c}{2}$ 
// find  $\theta(u) \forall u \in U$  by peeling
6  $\theta(u) \leftarrow -1 \forall u \in U$  // tip numbers
7 for each  $u$  with minimum  $\beta(u)$  s.t.  $\theta(u)$  is -1 do
8    $\theta(u) \leftarrow \beta(u)$ 
9   Find set  $\mathcal{B}$  of  $\mathbf{X}$ s containing  $u$ 
10  for each  $\mathbf{X} \in \mathcal{B}$  do
11     $x \leftarrow \text{other vertex in } \mathbf{X} \text{ s.t. } \theta(x) \text{ is } -1 (x \in U)$ 
12    if  $\beta(x) > \beta(u)$  then  $\beta(x) \leftarrow \beta(x) - 1$ 
13 return array  $\theta(\cdot)$ 

```

---

of butterflies that each vertex resides in. Then, we apply the peeling process; decrementing the butterfly count of the neighbor primary vertex at each step. Note that, there is only one such vertex in each butterfly.

Algorithm 1 finds the tip numbers of all the primary vertices. To find the  $k$ -tips and constructing the hierarchy on-the-fly, we use the disjoint-set forest and traversal avoidance heuristics that we introduced in a recent work [45]. Their application to the tip decomposition is straightforward and we do not give the details for brevity.

We aim to find the tip numbers of primary vertices, denoted as  $\beta(\cdot)$  in line 1. Lines 2 to 5 determine the number of butterflies for each  $u \in U$ . To do that, we combine the distance-2 neighbors of  $u$  in list  $D$  (line 3), and detect the counts of duplicates for each  $d \in D$ , except  $u$  since it is already in the butterflies we are looking for (line 4). This can be done in linear time by using a hashmap. Number of duplicates,  $c$ , for a vertex  $d \in D$  tells that how many secondary vertices in  $N(u)$  are connected to that vertex. Thus, we find the butterflies consisting  $u, d \in U$  and  $v_i, v_j \in V$  ( $i, j \leq c$ ), where  $v_1, v_2, \dots, v_c \in N(u)$  are connected to  $d$ , as shown in Figure 5. Since there are  $\binom{c}{2}$  pairs of  $(v_i, v_j)$ , we just compute the sum of  $\binom{c}{2}$  for each  $c \in C$ , where  $C$  is list of counts of duplicates for all  $d \in D$ .

In the peeling process (lines 7 to 12), we assign the tip number of vertices in the nondecreasing order of their butterfly counts. We leverage the bucket data structure to efficiently retrieve the vertex with the minimum number of butterflies. At each step, we first assign the current butterfly number of  $u$  as its tip number (line 8). Then, we find the butterflies that contains  $u$ , and in each butterfly we check the other primary vertex  $x \in U$ . If current butterfly number of  $x$  is greater than the current butterfly number of  $u$ , we decrement it since that butterfly will not contribute to the tip number of  $x$ .

The runtime of TIPDECOMPOSITION algorithm is defined by the time of the butterfly counting phase. The peeling phase takes only linear time, which is a lower bound for the first phase. Thus, time complexity is  $O(|\mathbf{X}_G|)$ , where  $\mathbf{X}_G$  is the set of butterflies in  $G$ . Butterfly counting can be speed up by using various algorithmic tricks, which are beyond the scope of this paper. Regarding the space complexity, all of the data structures are in at most  $O(|U|)$  size.

## 4.2 Wing Decomposition

We follow a similar path for the wing decomposition. Instead of looking at vertex-butterfly relations, we investigate the involvements of edges in butterflies. There are again two phases; counting the butterflies for each edge, and the peeling process to find wing numbers of edges. Just like the tip decomposition, we do not give the details about constructing  $k$ -wings and building the hierarchy for brevity. Algorithms in [45] are applicable for  $k$ -wings as well.

Algorithm 2 outlines the wing decomposition. Lines 2 to 6 counts the butterflies for each edge. Here, we utilize the ordering of vertices in  $U$  for efficient computation. All the vertices  $u \in U$  are processed in order (line 2), and in each intersection operation, we only take the vertices that succeed  $u$  (line 4). This way, each butterfly in  $G$  is visited only once.

In the peeling phase, wing numbers of edges are assigned in the nondecreasing order of their butterfly counts. There are four edges in a butterfly, and we need to check those three neighbors. In order to ensure that the butterfly that contains  $e$  is not examined before, we check if any of those three neighbor edges has been assigned a wing number yet (line 13). If they are all unassigned, we decrement their butterfly numbers if greater than the wing number we assigned in that step.

WINGDECOMPOSITION has  $O(|\mathbf{X}_G|)$  time complexity, but expected to take more time than TIPDECOMPOSITION due to constant factors in peeling phase. Additional space complexity of wing decomposition is  $O(|E|)$ , since we store butterfly numbers and wing numbers for each edge.

## 5. RELATED WORK

Bipartite graphs have been used to model the real-world networks with two different types of entities. There are many studies that showed the benefit of working on the bipartite model over the projected unipartite structure, and also some others advocating that the projections do not lose information and easier to analyze. The very first work that analyzed a bipartite network is introduced by Davis *et al.* [18] where they study a small network of women and their affiliations with clubs. Borgatti and Everett [12] redefined centrality and density metrics for bipartite graphs. They proposed to use bicliques to model dense regions. Robins and Alexander [43] analyzed the structure of interlocking directorates on raw bipartite data, and defined the clustering coefficients for bipartite networks. They proposed to use the ratio of 4-cycles in all 3-paths,  $\frac{4*|\mathbf{X}|}{|\mathbf{S}|}$ , which is analogous to the ratio of triangles to wedges in unipartite networks ( $\frac{3*|\Delta|}{|\Lambda|}$ ). Latapy *et al.* [31] presented a comprehensive summary of the literature on bipartite networks. They evaluated the previously defined statistics on real-world large bipartite networks with millions of edges. Their work is the first one to analyze large networks in contrast to the previous studies that focused on a few hundred nodes. Working on the bipartite network, instead of projection, is also useful for matrix partitioning [14] and clustering [49] algorithms. Despite all the attempts to redefine graph metrics and algorithms for bipartite networks, there are still some studies that propose to use projections for certain domains. Newman introduced the weighted projection for scientific collaboration networks [38]. Everett and Borgatti proposed to use dual projections [20]. The idea is to create projections for both set of nodes, and

---

### Algorithm 2: WINGDECOMPOSITION( $G = (U, V, E)$ )

---

```

// find the number of  $\mathbf{X}$  for each edge  $e \in E$ 
1  $\beta(e) \leftarrow 0$ , butterfly counts  $\forall e \in E$ 
2 for each  $u \in U$  in order do
3   for each  $v_1, v_2$  pair  $\in V$  do
4      $I \leftarrow N(v_1) \cap N(v_2)$  s.t.  $i \succ u \ \forall i \in I$ 
5     for each  $i \in I$  do
6        $\beta(e) \leftarrow \beta(e) + 1, \forall e \in \mathbf{X}(u, i, v_1, v_2)$ 

// find  $\psi(e) \ \forall e \in E$  by peeling
7  $\psi(e) \leftarrow -1 \ \forall e \in E$  // wing numbers
8 for each  $e$  with minimum  $\beta(e)$  s.t.  $\psi(e)$  is -1 do
9    $\psi(e) \leftarrow \beta(e)$ 
10  Find set  $\mathcal{B}$  of  $\mathbf{X}$ s containing  $e$ 
11  for each  $\mathbf{X} \in \mathcal{B}$  do
12     $F \leftarrow$  other three edges in  $\mathbf{X}$ 
13    if  $\psi(f)$  is -1  $\forall f \in F$  then
14      for each  $f \in F$  do
15        if  $\beta(f) > \beta(e)$  then  $\beta(f) \leftarrow \beta(f) - 1$ 

16 return array  $\psi(\cdot)$ 

```

---

use the resulting one-mode networks for analysis. We focus on using the actual bipartite data (without projection) to find many dense subgraphs and their relations to each other, which have not been explored thoroughly.

**Applications:** Dense subgraphs in bipartite networks have valuable information that can be utilized for various application domains. Newman [38, 39] studied author and paper relations on coauthorship networks. He applied different projections to convert the bipartite network to a unipartite network. Giatsidis *et al.* [21] worked on the same problem by using the same projection technique. In addition to finding dense regions, they also focused on detecting the hierarchy and adapted the  $k$ -core decomposition for weighted networks. They defined the **fractional  $k$ -core** as a maximal subgraph where every vertex has at least weight  $k$ . Another domain that bipartite dense subgraphs are useful for is fraud detection. Fake likes, ratings and reviews are prevalent in online social networks where users interact with pages, posts etc. Those interactions can be modeled as a bipartite network. Beutel *et al.* [11] studied the Facebook network to detect fake likes. They model this problem as finding the nearly complete temporal bicliques. More recently, Hooi *et al.* [27] worked on user-product and follower-followee networks to detect fraudulent behavior. They used average bipartite degree optimization to find fake reviews/follows even when the fraudsters camouflage themselves by liking/reviewing random pages.

**Bipartite dense subgraphs:** Borgatti and Everett proposed biclique to model dense subgraphs [12] which is a fully connected subgraph between two set of nodes. Kumar *et al.* used bicliques of various sizes to analyze web graphs [29]. Enumerating all the maximal bicliques and quasi-cliques is studied by Sim *et al.* [48], and Mukherjee and Tirthapura [37].

However, biclique definition is too strict that does not tolerate even a single missing edge. More recently, Tsourakakis *et al.* [36] used sampling to find  $(p, q)$ -biclique densest sub-

graph in bipartite networks. They used yet another definition of density which considers the number of  $(p, q)$ -bicliques over all possible ones, and find the densest subgraph accordingly. Our algorithms do not focus on finding only a single subgraph that is perfectly dense. Instead, main motivation is to find a detailed hierarchy that consists of many dense subgraphs.

**Bipartite community detection:** Literature is quite rich on community detection for unipartite networks. Certain metrics are adapted for bipartite networks to find high quality communities. Guimera *et al.* [25] refined the modularity, inspired from Newman’s influential paper [41]. Barber and Clark [8] adapted the label propagation algorithm for bipartite networks to optimize the modularity, which has linear complexity in the number of edges, but gives communities with moderate quality. On a related direction, Larremore *et al.* [30] proposed a bipartite stochastic block model for community detection in bipartite networks, which needs the resulting number of communities a priori. More information can be found in [5] where Alzahrani and Horadam survey the community detection works on bipartite networks. Main difference of our work is that we do not optimize any metric to find the best region. Our peeling algorithms are deterministic, provide many dense subgraphs with hierarchical relations, and only take the graph as input.

**Peeling on bipartite networks:** There have been some attempts to adapt peeling algorithms or  $k$ -core [47] like subgraphs to the bipartite networks. Cerinsek and Batagelj [15] adapted the generalized core idea [9] to bipartite networks. They define the  $(p, q)$ -core with monotonic  $f$  and  $g$  functions as a maximal union of the set of vertices  $u \in U$  s.t.  $f(u) \geq p$  and  $v \in V$  s.t.  $g(v) \geq q$ . However, their definition is not suitable to construct a hierarchy among  $(p, q)$ -cores since it is not clear how to define comparison function for  $(p, q)$  pairs. Another related work is done by Li *et al.* [34] where they adapt the  $k$ -truss like definition for bipartite networks. They insert artificial edges between vertex pairs that are in the same side and sharing a neighbor. Then they apply peeling algorithm on those artificial edges and their triangle counts. This is actually identical to creating the projection, and applying the  $k$ -truss decomposition using the triangle counts [17].

## 6. EXPERIMENTS

We evaluate our algorithms and existing previous alternatives on different kinds of real-world bipartite networks, obtained from SNAP [32] and ICON [2]. **condmat** is the author-paper network for the arXiv preprints in condensed matter physics between 1995 and 1999 [40]. **dbconf** is another author-paper network that we constructed with the papers and the authors from three top database conferences; VLDB, SIGMOD, and ICDE [33]. **github** is the network between users and repositories in the GitHub, released in the 2009 GitHub Contest [16]. **marvel** is occurrence relations between the Marvel characters and the comic books [4]. **IMDb** links actors and the movies they played in [1], and **lastfm** is the network of users and the artists they listened in Last.fm online music system [13]. Table 1 gives some important statistics. In second to fourth columns, we show the number of primary vertices, secondary vertices, and edges in each network. We assume that the primary vertices are the ones

Table 1: Statistics for the real-world graphs.

network	$ U $	$ V $	$ E $	$ E_p $	$ \mathbf{X} $	$ \Delta_p $
condmat	16.7K	22.0K	58.6K	95.1K	70.5K	68.0K
dbconf	11.2K	8.9K	30.7K	84.8K	34.6K	95.7K
github	56.6K	123.3K	440.2K	44.5M	50.9M	962.6M
marvel	6.5K	12.9K	96.7K	336.5K	10.7M	3.3M
IMDb	1, 2M	419.7K	5.6M	157.5M	42.5M	312.1M
lastfm	2.1K	18.7K	92.8K	2.0M	13.4M	296.1M

that drive the connections. In the fifth column, number of edges in the projected graph ( $E_p$ ) with respect to the primary vertices are given (Definition 1). Last two columns show the butterfly ( $\mathbf{X}$ ) counts in each bipartite network and the triangle ( $\Delta_p$ ) counts in the projected unipartite graph. Algorithms are implemented in C++ and compiled using gcc 5.2.0 at -O2 optimization level. All experiments are performed on a Linux operating system running on a machine with Intel Xeon Haswell E5-2698 2.30 GHz processor with 128 GB of RAM.

We compared the TIPDECOMPOSITION (Algorithm 1) and WINGDECOMPOSITION (Algorithm 2) with the previous works that find many dense subgraphs with an hierarchy on the bipartite graph or its unipartite (un)weighted projection:

- For the **unweighted projection**, we use two nucleus decomposition algorithms which find the  $k$ -cores and  $(2, 3)$ -nuclei (Definition 3 and 4).  $(2, 3)$ -nuclei has been shown to be quite effective to find dense regions [28, 46] with hierarchical relations.
- On the **weighted projection**, we find the fractional  $k$ -cores [21]. It is the only peeling adaptation for weighted networks, to the best of our knowledge.
- Regarding the **bipartite data**, Li *et al.* [34] proposed a  $k$ -truss adaptation for bipartite networks, as explained in the last part of previous section. Although the focus is on the bipartite connections, their algorithm relies on inserting edges between the vertices in the same set, and compute the  $k$ -trusses on those new edges and new triangles, which is essentially same as the  $(2, 3)$  nucleus decomposition.

We report the bipartite subgraphs by their sizes in primary and secondary vertex sets, and the edge density, i.e.  $\frac{|E|}{|U| \cdot |V|}$ . For the nucleus decomposition and fractional  $k$ -core algorithms, we find the nuclei/cores in the unipartite graph ( $G_p$ ), and then report the induced bipartite subgraphs using (primary) vertices in those nuclei/cores.

We first compare the dense subgraph profiles, then introduce some use cases that highlight some interesting findings in the **dbconf** network, and finish by checking the runtime results.

### 6.1 Dense subgraph profiles

We compare the size and density of bipartite subgraphs obtained by  $k$ -wings,  $k$ -tips,  $k$ -cores,  $(2, 3)$ -nuclei, and fractional  $k$ -cores. Figures 6 (**condmat**), 7 (**dbconf**), 8 (**github**), 9 (**marvel**), and also 1 (**IMDb**) (in Section 1) summarize the results with informational captions where each dot is a bipartite subgraph with at least 0.1 density and at most 10,000 vertices in either side. Primary and secondary vertex set sizes are given on the x- and y-axis, respectively, and the density is color coded. We omit the results for  $k$ -cores since



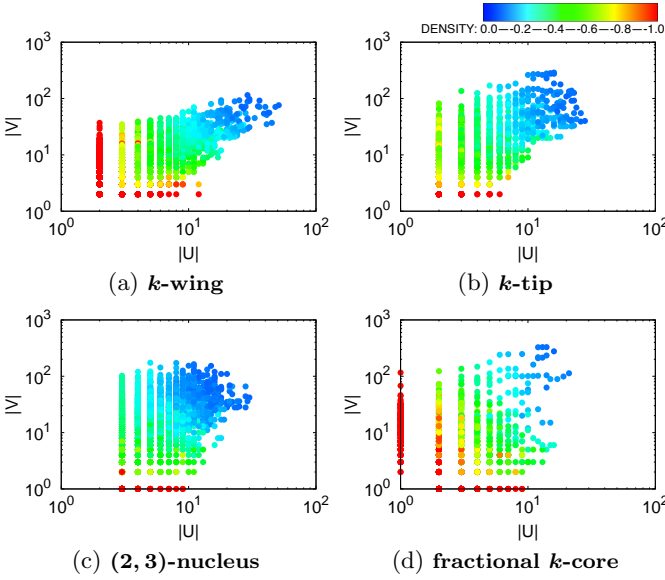


Figure 6: Dense subgraphs in *condmat*. *k-wing* has 4820 subgraphs with  $\geq 0.1$  density. 416 of them have sizes of  $\geq 5$  on both sides with  $\geq 0.5$  density, whereas this number is 59, 14, and 20 in *k-tip*, (2,3)-nucleus and fractional *k-core* cases.

they are consistently worse than the (2,3)-nuclei. Overall, we observe that dense bipartite subgraphs with nontrivial sizes on both sides can be obtained with *k-wings*. *k-tips* also perform well on a few instances compared to the others, but not as good as the *k-wings*. In particular, 33% of the *k-wing* subgraphs in *IMDb*, for instance, have  $\geq 5$  vertices on each side with  $\geq 0.5$  density. On *lastfm* network, only *k-wings* can output subgraphs with  $> 0.1$  density that has less than 10,000 vertices. Furthermore, 117 of them have  $\geq 20$  vertices on each side with at least 0.5 density.

We also observe dense structures in (2,3)-nuclei and fractional *k-cores* where the secondary vertex size is just 1 (red dots along x-axes). Those represent the involvements of multiple (primary) entities in the same (secondary) event **for only one time**. In *dbconf*, those are mostly the product papers authored by a group of researchers in a company, and for the most cases this is the only paper authors have written, not implying a significance. For example “Comdb2 Bloomberg’s Highly Available Relational Database System” paper in ICDE’10 is written by a group of researchers in Bloomberg LP. On the other hand, there are also instances where the red dots appear along y-axis in fractional *k-core* cases. Again in *dbconf* network, for instance, those are the prolific authors that are actually very close to many dense structures, but isolated due to the strict stepping semantic in the fractional *k-core* computation. Divesh Srivastava is one such example with 144 papers who is not reported in any other subgraph with more than 0.5 density.

## 6.2 Peeling the top database conferences

In this section, we highlight some interesting subgraphs and hierarchical structures in the *dbconf* network. We focus on what our algorithms can find that cannot be obtained with other algorithms, and vice versa.

**DB and Games at Cornell:** One of the leaf subgraphs in the hierarchy tree given by *k-wings* contains some professors and students from Cornell University and the papers they published. It has 6 authors and 11 papers with 0.7 den-

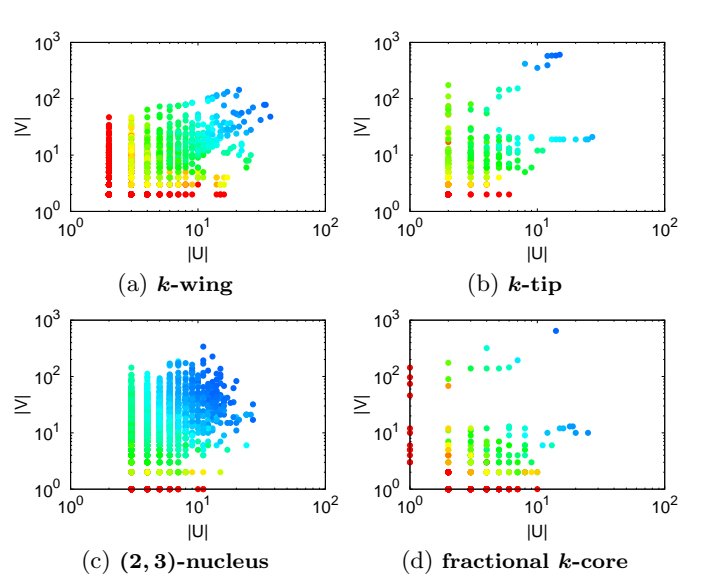


Figure 7: *dbconf* network. There are 4 *k-wings* with  $\geq 10$  vertices on both sides with  $\geq 0.4$  density. No such subgraphs exist in others.

sity. We noticed that one of the authors is a game design researcher, W. M. White<sup>1</sup>, and the papers are about designing event processing systems for the massively multiplayer online games. Other authors are J. Gehrke and A. J. Demers (database professors), M. A. Vaz Salles (ex-postdoc worked with J. Gehrke), T. Cao and B. Sowell (ex-students of J. Gehrke). There are three papers (VLDB09, VLDB10, SIGMOD11) that has all 6 authors. Each author contributed to at least 5 papers, and any pair of authors published at least 4 papers together. On the other hand, *k-tips*, *k-cores*, and fractional *k-cores* cannot find any subgraph with more than 0.1 density that contains W. M. White, who published almost all of his papers with J. Gehrke on game research. (2,3)-nucleus finds a subgraph with 15 authors, including W. M. White, with 0.11 density. This set of authors does

<sup>1</sup><https://www.cs.cornell.edu/~wmwhite/>

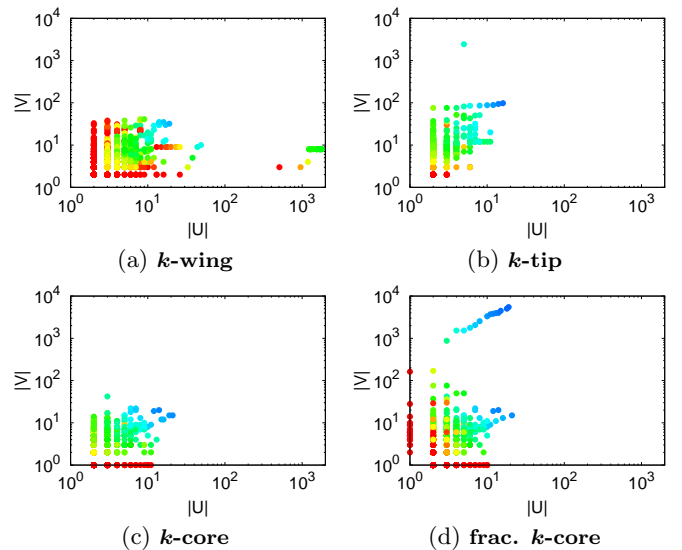


Figure 8: Dense structures in *github* network. *k-wings* can find 27 structures with  $\geq 5$  vertices on both sides and  $\geq 0.9$  density where other algorithms cannot. *k-tips* and fractional *k-cores* show similar performances.

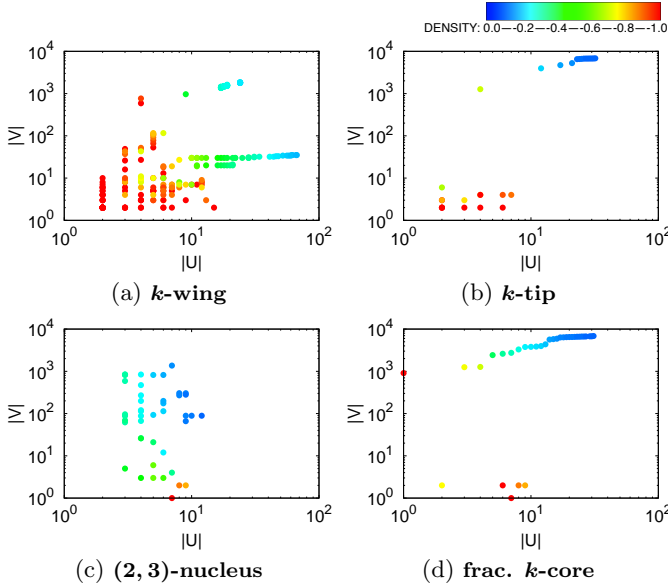


Figure 9: Results for *marvel* network. There are 11 subgraphs with at least 10 vertices on each side and 0.5 density in the  $k$ -wing case. No such subgraphs can be obtained with the others. not contain T. Cao and B. Sowell, who published three papers with W. M. White, but includes seven authors who only collaborated on one paper.

**H. Ishikawa and coauthors:** On the other side, we checked the densest non-trivial bipartite subgraphs given by (2, 3)-nuclei. One of those has 10 authors, 5 papers, and 0.42 density. However, most authors have only one paper in the subgraph. H. Ishikawa is the author with most papers among them, and we checked his subgraphs in  $k$ -wings. We found a biclique with 4 authors and 3 papers, and also a fifth author is included with the same papers in a subgraph with 0.93 density.

**Philip Yu in leaves:** Lastly, we checked the densest subgraphs at the bottom of the hierarchy (leaves) that contain Philip Yu, a prolific researcher. Tracing the paths from those leaves to the upper levels gives interesting insights, illustrated in Figure 10. Leaves are on the right (in red), and upper levels are going to left with decreasing densities. Tree on the bottom has researchers from IBM Research Streams group, and their close collaborators. In the bottom branch, K. L. Wu (manager) and P. Yu (ex-manager) is shown as a leaf which is a biclique with their 13 papers. Then B. Gedik joins them in a 0.82 density subgraph, who used to be a staff member in the same group. B. Gedik’s PhD advisor, Ling Liu, appears on the upper levels, and also the other ex-IBM researchers H. Andrade and G. Jacques-Silva. Regarding the other branch, C. Aggarwal (staff member) forms another biclique with 14 papers. People on its upper levels are also (ex) IBM researchers. Root of the bottom tree has 76 researchers and interns from the same group with 0.16 density.

The other tree on the top shows other connections with other prolific researchers Jiawei Han, Xifeng Yan, Haixun Wang, and Wei Wang, some of whom also had worked in the same group at IBM.

### 6.3 Runtime performances

At the end, we check the runtime results. As explained in Section 4, TIPDECOMPOSITION and WINGDECOMPOSITION

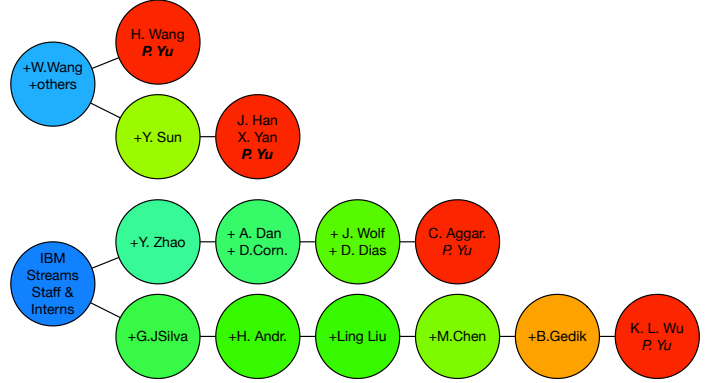


Figure 10:  $k$ -wings in *dbconf* network that contains Philip Yu. Subgraphs densities are color coded. The bottom tree shows the collaborations in IBM Research’s Streams group. There are also some people who worked in the same group on the top tree.

has  $O(|\mathbf{X}|)$  complexity, where the latter is expected to be slower due to hidden constants. (2, 3)-nucleus has  $O(|\Delta|)$ , and fractional  $k$ -core has  $O(|E|)$  time complexities. Table 2 shows the total runtimes for all algorithms. Results verify the complexities;  $k$ -wing is orders of magnitude faster than (2, 3)-nuclei on *github*, *IMDB*, and *lastfm* networks, where (2, 3)-nucleus on *github* is not finished in two days. Note that those three networks have more triangles in their projections than the butterflies in their bipartite forms (Table 1). On the other hand, *marvel* network has more butterflies, and WINGDECOMPOSITION takes more time.

(in seconds)	bipartite		unipartite	
	$k$ -tip	$k$ -wing	(2, 3)-nucleus	fractional $k$ -core
condmat	0.04	0.13	0.05	0.04
dbconf	0.04	0.07	0.06	0.03
github	29.93	118.41	> 2 days	9.70
marvel	3.90	19.22	3.15	0.12
IMDB	66.07	170.50	39258.34	37.27
lastfm	3.59	20.95	767.22	0.21

Table 2: Runtime performances of the algorithms.  $k$ -wing is orders of magnitude faster than (2, 3)-nucleus decomposition on the large networks.

## 7. SUMMARY AND OPEN QUESTIONS

We proposed peeling algorithms for bipartite networks to find many dense substructures and a hierarchy among them. Our peeling algorithms are based on the butterfly subgraphs, and works on the involvements of vertices ( $k$ -tip) and edges ( $k$ -wing) in butterfly structures. Experiments and use case analyses verify the quality and runtime performance of our contributions with respect to previous works.

As a future work, we plan to speed up the computation by shared-memory parallelism, especially for the cases when the graph has too many butterflies. We also want to explore further higher-order structures in bipartite networks that can give more insights about the network and also easy to compute.

## 8. ACKNOWLEDGEMENTS

Sandia National Laboratories is a multi-mission laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy’s National Nuclear Security Administration under contract DE-AC04-94AL85000.



## 9. REFERENCES

- [1] Imdb. ([www.imdb.com/interfaces](http://www.imdb.com/interfaces)).
- [2] E. T. Aaron Clauset and M. Sainz. The colorado index of complex networks., 2016. ([icon.colorado.edu](http://icon.colorado.edu)).
- [3] S. Aksoy, T. G. Kolda, and A. Pinar. Measuring and modeling bipartite graphs with community structure. *CoRR*, abs/1607.08673, 2016.
- [4] R. Alberich, J. Miro-Julia, and F. Rosselló. Marvel universe looks almost like a real social network. Preprint, 2002.
- [5] T. Alzahrani and K. J. Horadam. *Community Detection in Bipartite Networks: Algorithms and Case studies*, pages 25–50. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016.
- [6] A. Angel, N. Sarkas, N. Koudas, and D. Srivastava. Dense subgraph maintenance under streaming edge weight updates for real-time story identification. *PVLDB*, 5(6):574–585, Feb. 2012.
- [7] B. Balasundaram, S. Butenko, and I. Hicks. Clique relaxations in social network analysis: The maximum  $k$ -plex problem. 59:133–142, 2011.
- [8] M. J. Barber and J. W. Clark. Detecting network communities by propagating labels under constraints. *Phys. Rev. E*, 80:026129, 2009.
- [9] V. Batagelj and M. Zaversnik. Generalized cores. Technical report, ArXiv, 2002.
- [10] A. R. Benson, D. F. Gleich, and J. Leskovec. Higher-order organization of complex networks. *Science*, 353(6295):163–166, 2016.
- [11] A. Beutel, W. Xu, V. Guruswami, C. Palow, and C. Faloutsos. Copycatch: Stopping group attacks by spotting lockstep behavior in social networks. In *Proceedings of the 22Nd International Conference on World Wide Web, WWW '13*, pages 119–130, 2013.
- [12] S. P. Borgatti and M. G. Everett. Network analysis of 2-mode data. *Social Networks*, 19(3):243 – 269, 1997.
- [13] I. Cantador, P. Brusilovsky, and T. Kuflik. 2nd workshop on information heterogeneity and fusion in recommender systems (hetrec 2011). In *Proceedings of the 5th ACM conference on Recommender systems, RecSys 2011*, New York, NY, USA, 2011. ACM.
- [14] Ü. V. Çatalyürek and C. Aykanat. Hypergraph-partitioning-based decomposition for parallel sparse-matrix vector multiplication. *IEEE Transactions on Parallel and Distributed Systems*, 10(7):673–693, 1999.
- [15] M. Cerinsek and V. Batagelj. Generalized two-mode cores. *Social Networks*, 42:80 – 87, 2015.
- [16] S. Chacon. The 2009 github contest. ([github.com/blog/466-the-2009-github-contest](https://github.com/blog/466-the-2009-github-contest)).
- [17] J. Cohen. Trusses: Cohesive subgraphs for social network analysis. National Security Agency Technical Report, 2008.
- [18] A. Davis, B. B. Gardner, M. R. Gardner, and W. L. Warner. *Deep South: A Social Anthropological Study Of Caste And Class*. The University of Chicago Press, Chicago, Ill., 1941.
- [19] X. Du, R. Jin, L. Ding, V. E. Lee, and J. H. T. Jr. Migration motif: a spatial - temporal pattern mining approach for financial markets. In *Proc. of the 15th ACM SIGKDD International Conf. on Knowledge Discovery and Data Mining*, pages 1135–1144, 2009.
- [20] M. Everett and S. Borgatti. The dual-projection approach for two-mode networks. *Social Networks*, 35(2):204 – 210, 2013.
- [21] C. Giatsidis, D. M. Thilikos, and M. Vazirgiannis. Evaluating cooperation in communities with the  $k$ -core structure. In *ASONAM*, pages 87–93, 2011.
- [22] D. Gibson, R. Kumar, and A. Tomkins. Discovering large dense subgraphs in massive graphs. In *VLDB*, pages 721–732, 2005.
- [23] A. Gionis, F. Junqueira, V. Leroy, M. Serafini, and I. Weber. Piggybacking on social networks. *Proc. VLDB Endow.*, 6(6):409–420, 2013.
- [24] E. Gregori, L. Lenzini, and C. Orsini.  $k$ -dense communities in the internet as-level topology. In *International Conf. on Communication Systems and Networks (COMSNETS)*, pages 1–10, 2011.
- [25] R. Guimera, M. Sales-Pardo, and L. A. N. Amaral. Module identification in bipartite and directed networks. *Phys. Rev. E*, 76:036102, 2007.
- [26] J. Healy, J. Janssen, E. Milios, and W. Aiello. *Characterization of Graphs Using Degree Cores*, pages 137–148. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.
- [27] B. Hooi, H. A. Song, A. Beutel, N. Shah, K. Shin, and C. Faloutsos. Fraudster: Bounding graph fraud in the face of camouflage. In *KDD*, pages 895–904, 2016.
- [28] X. Huang, H. Cheng, L. Qin, W. Tian, and J. X. Yu. Querying  $k$ -truss community in large and dynamic graphs. In *Proc. of the ACM SIGMOD International Conf. on Management of Data*, pages 1311–1322, 2014.
- [29] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Trawling the web for emerging cyber-communities. In *WWW*, pages 1481–1493, 1999.
- [30] D. B. Larremore, A. Clauset, and A. Z. Jacobs. Efficiently inferring community structure in bipartite networks. *Phys. Rev. E*, 90:012805, Jul 2014.
- [31] M. Latapy, C. Magnien, and N. D. Vecchio. Basic notions for the analysis of large two-mode networks. *Social Networks*, 30(1):31 – 48, 2008.
- [32] J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection, June 2014. ([snap.stanford.edu/data](http://snap.stanford.edu/data)).
- [33] M. Ley. Dblp computer science bibliography, Sept. 2016. ([dblp.uni-trier.de](http://dblp.uni-trier.de)).
- [34] Y. Li, T. Kuboyama, and H. Sakamoto. Truss decomposition for extracting communities in bipartite graph. In *IMMM 2013 : The Third International Conference on Advances in Information Mining and Management*, 2013.
- [35] D. Matula and L. Beck. Smallest-last ordering and clustering and graph coloring algorithms. *Journal of ACM*, 30(3):417–427, 1983.
- [36] M. Mitzenmacher, J. Pachocki, R. Peng, C. Tsourakakis, and S. C. Xu. Scalable large near-clique detection in large-scale networks via sampling. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '15*, pages 815–824, 2015.
- [37] A. P. Mukherjee and S. Tirthapura. Enumerating maximal bicliques from a large graph using mapreduce. In *Proceedings of the 2014 IEEE International Congress on Big Data, BIGDATAACONGRESS '14*, pages 707–716, 2014.
- [38] M. E. J. Newman. Scientific collaboration networks. i. network construction and fundamental results. *Phys. Rev. E*, 64:016131, 2001.
- [39] M. E. J. Newman. Scientific collaboration networks. ii. shortest paths, weighted networks, and centrality. *Phys. Rev. E*, 64:016132, 2001.
- [40] M. E. J. Newman. The structure of scientific collaboration networks. *Proceedings of the National Academy of Sciences*, 98(2):404–409, 2001.
- [41] M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Phys. Rev. E*, 74:036104, 2006.
- [42] T. Opsahl. Triadic closure in two-mode networks: Redefining the global and local clustering coefficients. *Social Networks*, 35(2):159 – 167, 2013. Special Issue on Advances in Two-mode Social Networks.
- [43] G. Robins and M. Alexander. Small worlds among interlocking directors: Network structure and distance in bipartite graphs. *Computational & Mathematical Organization Theory*, 10(1):69–94, 2004.
- [44] K. Saito and T. Yamada. Extracting communities from complex networks by the  $k$ -dense method. In *IEEE International Conf. on Data Mining Workshops, ICDMW*, pages 300–304, 2006.
- [45] A. E. Sariyüce and A. Pinar. Fast hierarchy construction for dense subgraphs. *Proc. VLDB Endow.*, 2017. to appear.
- [46] A. E. Sariyüce, C. Seshadhri, A. Pinar, and Ü. V. Çatalyürek. Finding the hierarchy of dense subgraphs using nucleus decompositions. In *Proc. of the International Conf. on World Wide Web (WWW)*, pages 927–937, 2015.
- [47] S. B. Seidman. Network structure and minimum degree. *Social Networks*, 5(3):269–287, 1983.
- [48] K. Sim, J. Li, V. Gopalkrishnan, and G. Liu. Mining maximal quasi-bicliques: Novel algorithm and applications in the stock market and protein networks. *Statistical Analysis and Data Mining*, 2(4):255–273, 2009.
- [49] M. M. Wolf, A. M. Klinvex, and D. M. Dunlavy. Advantages to modeling relational data using hypergraphs versus graphs. In *IEEE High Performance Extreme Computing Conference, HPEC*, 2016.
- [50] B. Zhang and S. Horvath. A general framework for weighted gene co-expression network analysis. *Statistical Applications in Genetics and Molecular Biology*, 4(1):Article 17+, 2005.
- [51] Y. Zhang and S. Parthasarathy. Extracting analyzing and visualizing triangle  $k$ -core motifs within networks. In *Proc. of the IEEE International Conf. on Data Engineering, ICDE*, pages 1049–1060, 2012.